Home Credit Default Risk

Can you predict how capable each applicant is of repaying a loan?

Prepared By: Soumee Ghosh (2211444)

Dataset Description

Many people struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, this population is often taken advantage of by untrustworthy lenders.

Home Credit strives to broaden financial inclusion for the unbanked population by providing a positive and safe borrowing experience. In order to make sure this underserved population has a positive loan experience, Home Credit makes use of a variety of alternative data--including telco and transactional information--to predict their clients' repayment abilities.

While Home Credit is currently using various statistical and machine learning methods to make these predictions, they're challenging Kagglers to help them unlock the full potential of their data. Doing so will ensure that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful.

Problem:

The Description mention that the goal of this compition is to filter out people whether they would pay the loan transaction on time or not,

giving that you have very small amount of data for those people (People aren't have historical transaction).

Target variable

1 - client with payment difficulties: he/she had late payment more than X days on at least one of the first Y installments of the loan in our sample.

0 - all other cases

The analysis can be done in the following steps:

- Understanding the data
- EDA and Data Cleaning
- Data Pre-Processing
- Model building & Evaluation
- Hyperparameter Tuning
- Prediction.

Firstly we import the necessary libraries and load the application train and test dataset and check for the shape of the data.

We find that the training set consists of 307511 rows and 122 columns whereas the testing set consists of 48744 rows and 121 columns.

Next we move to EDA and data cleaning.

Under EDA we perform the following operations:

* Univariate Analysis

  * 1.1 Inspect for duplicates

  * 1.2 Inspect NaN values

  * 1.3 Inspect for Unreal data and transform it to NaN (if needed)

  * 1.4 Inspect numerical and categorical features for transformations (If needed)

  * 1.5 Inspect Outliers

  * 1.6 Inspect all the features and see whether there is a need to change data types or need to special transformation (e.g. log transformation) or is there an imbalanced data.

Firstly on checking for null values we find that there is a lot of null values in the train as well as test set.

Next we filter only the categorical variables and check for unique values. We can observe that code_gender and organization_type columns have XNA and XNP which we can consider as NAN values. Also we observe that NAME_FAMILY_STATUS contains a value as ‘Uknown’ which we will consider as NAN.

We then replace ‘XNA’ , ‘XNP’ and ‘Uknown’ by NAN.

Thus we get the total null values as:

```
[15]: print(app_train[['ORGANIZATION_TYPE','NAME_FAMILY_STATUS', 'CODE_GENDER']].isnull().sum())
      print('-'*40)
      print(app_test[['ORGANIZATION_TYPE','NAME_FAMILY_STATUS', 'CODE_GENDER']].isnull().sum())

      ORGANIZATION_TYPE     55374
      NAME_FAMILY_STATUS        2
      CODE_GENDER               4
      dtype: int64
      ----------------------------------------
      ORGANIZATION_TYPE     9274
      NAME_FAMILY_STATUS       0
      CODE_GENDER              0
      dtype: int64
```

Next we drop the columns that contain more than 40% NAN values.

Transformations:

For speeding the approach , columns which contain a very small amount of NAN values we will convert it to the most frequent i.e. Mode value in case of categorical variables and Mean in case of numerical variables. We will do this for the columns that contain NAN values below 14%.

Inspect outliers:

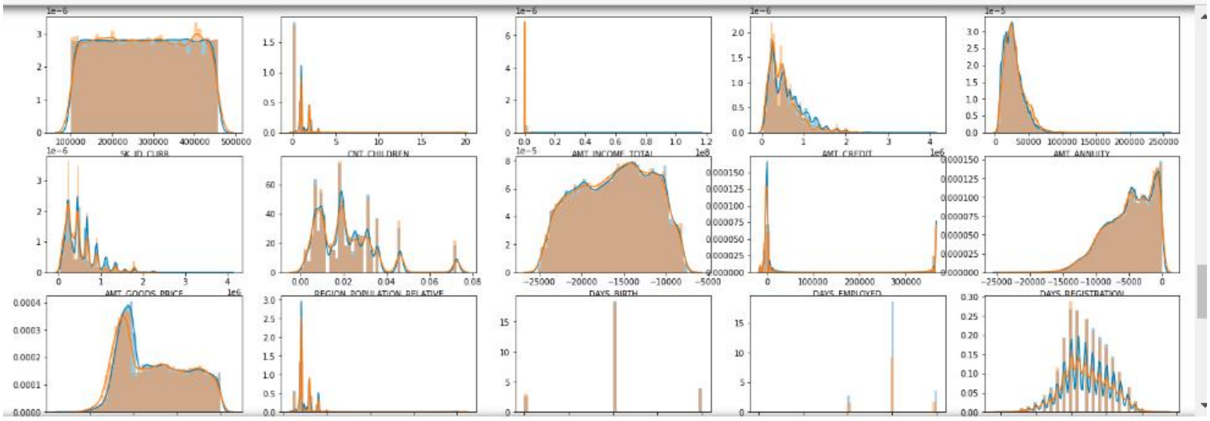
Inspecting for outliers is a very important step as we need to inspect for presence of outliers before we start imputing the missing or NAN values.

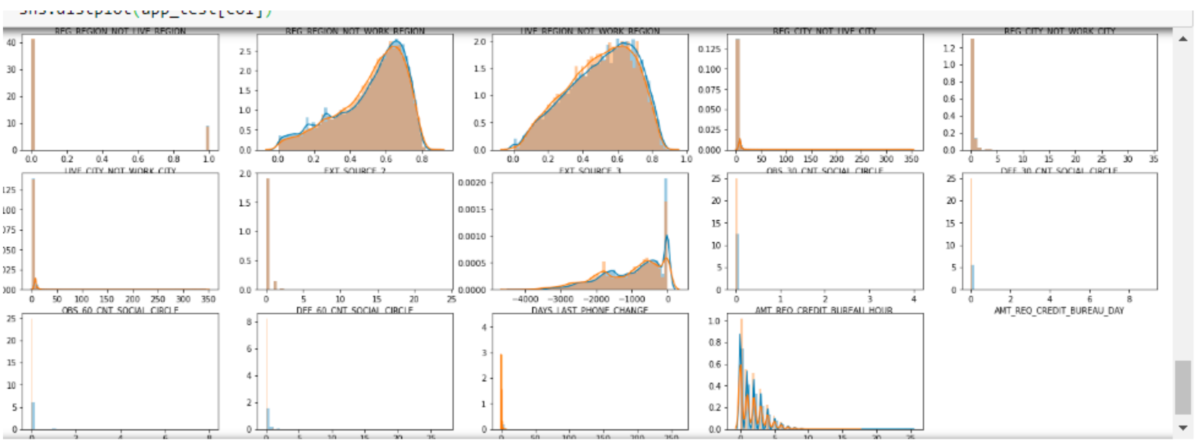
We see that some of the features are very skewed to the right and some aren't. So we will impute using mode for all variables except ext_source_2 and AMT_ANNUITY . For them we will use mean as from their distribution we can see that mean will be a reliable value.

For AMT_GOOD_PRICE we impute the null values using median.

Next we try extracting continuous columns i.e. numerical columns except the TARGET column and flags columns . We try to draw histogram for all the numerical columns to observe their distribution.

```
sns.distplot(app_test[col])
```





These graphs doesn't give much information but we need to dig deeper in every column but what we can conclude from here is that features of the train and test data have similar distributions and some features are there which we need to handle before proceeding ahead.

On observing the statistical description of the continuous columns we come across the following findings.

Findings

* Days are in negatives! it may be that the data source give all days data with negatives, simply we need to multiply it by -1

* DAYS_EMPLOYED have maximum value positive and it's 1000 years, also checking this link: https://www.kaggle.com/c/home-credit-default-risk/discussion/57248 the competition hosts mention that this value means infinity which we need to deal with.

It says in the competition documentation that DAYS_EMPLOYED is 'How many days before the application the person started current employment' and the minimum value for that column is 49 years so still in the same job 50 years!! is kind of suspicious.

* It seems to me that maximum income is very suspicious as it's a very huge number comparing to Loan amount (Why someone income is 25 pounds and ask for 1 pound loan!).

* Maximum Age is 69 years old! and maximum employing time is 49 years old! -- we need to check whether is there’s any one observation that has a number of employing days more than the number of birth days (Note: you will see the minimum in days is 69 years but as the values are negatives so we will consider as maximum)

We try to deal with these problems. We first deal with unreal value for days_employed and we convert all values with days employed as 365243 to NAN .

We will try to impute those NAN values using average employment time group by the occupation type.

Also the average years between the days_birth and Days_employed will depend on the occupation type.

```
app_train.groupby(['OCCUPATION_TYPE'])['DAYS_EMPLOYED'].mean()
```

```
2]: OCCUPATION_TYPE
    Accountants             -2394.102823
    Cleaning staff          -2131.155665
    Cooking staff           -2152.466868
    Core staff              -2797.755967
    Drivers                 -1939.034618
    HR staff                -2278.866785
    High skill tech staff   -2739.979086
    IT staff                -2095.570342
    Laborers                -2424.143152
    Low-skill Laborers      -1664.186813
    Managers                -2759.318937
    Medicine staff          -3750.265550
    Private service staff   -2238.281297
    Realty agents           -1785.003995
    Sales staff             -1703.789421
    Secretaries             -2607.050575
    Security staff          -1904.809106
    Waiters/barmen staff    -1873.172849
    Name: DAYS_EMPLOYED, dtype: float64
```

But before proceeding we first need to impute the nan values present in occupation type.

We observe that the maximum income of clients is about 30 times the maximum amount of loans so we look for the suspicious observations. We firstly create a dataframe where total income is greater than 1 Million and we check for credit amount, annuity amount, no of children and the target column.

Then we try to create a ratio of credit to income and annuity to income and look for only those clients that have difficulty paying the loan i.e. Target=1.

[43]:
```python
# Tha maximum income of a the clients is about 30 times the maximum amount of the loans

## create dataframe with total income > 1M
susp_df1 = app_train[app_train['AMT_INCOME_TOTAL']>1e+6][['AMT_INCOME_TOTAL','AMT_CREDIT','AMT_ANNUITY','CNT_CHILDREN', 'TARGET']

## create Credit/Income and Annuity/Income percentages
susp_df1['Credit/Income'] = susp_df1['AMT_CREDIT']/susp_df1['AMT_INCOME_TOTAL']
susp_df1['Annuity/Income'] = susp_df1['AMT_ANNUITY']/susp_df1['AMT_INCOME_TOTAL']

## show only clients with difficuties
susp_df1[susp_df1['TARGET']==1].sort_values(by='Credit/Income', ascending=True)
```

[43]:

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	CNT_CHILDREN	TARGET	Credit/Income	Annuity/Income
12840	117000000.0	562491.0	26194.5	1	1	0.004808	0.000224
248159	3150000.0	900000.0	48825.0	1	1	0.285714	0.015500
151018	1080000.0	371245.5	17437.5	0	1	0.343746	0.016146
167656	1575000.0	553806.0	28273.5	0	1	0.351623	0.017951
173663	1350000.0	491211.0	50463.0	3	1	0.363860	0.037380
41725	1890000.0	781920.0	61906.5	1	1	0.413714	0.032755
234728	1350000.0	576072.0	28017.0	0	1	0.426720	0.020753
248970	1890000.0	900000.0	57649.5	0	1	0.476190	0.030502
265884	1170000.0	983299.5	41661.0	1	1	0.840427	0.035608

The 1st record with credit to income ratio as less than 0.005 and income over 117 million is not logical So we drop this observation.

We observe that the maximum age of the client is 69 years so we extract a dataframe with only days of birth and the target column and we look only for those people who are aged above 65 years.

We find that among them 7588 people have no difficulty in paying the loan but 288 people face difficulty.

Now we try to analyse the categorical columns and observe the following.

* NAME_TYPE_SUIT : we will shorten the categories and merge Other_A, Other_B and Group of people to same Group called Others

* NAME_INCOME_TYPE: we will shorten the categories and merge Unemployed, student, Maternity leave to same group as logically all of them don't have source of money so will behave similarly regarding the target value

* ORGNIZATION_TYPE : we will take around 10-15 category and merge others to same group called Other

* we have more than 30% of OCCUPATION_TYPE in train and test data as NaNs, we will fill it using the mode grouping by the NAME_EDUCATION_TYPE because it's the most reasonable column who can refer to the occupation type

* we have more than 17% of EXT_SOURCE_3 in train and test data as NaNs, we will fill it using the mean grouping by the OCUPPATION_TYPE

* we have more than 18% of ORGNIZATION_TYPE in train and test data as NaNs, we will fill it using the mode grouping by the OCUPPATION_TYPE.

After all the cleaning and imputation we move to our target variable.

We find that thereis great imbalance in the target variable and class 1 is only 8 % of the data.

dex	TARGET
0	0.919274
1	0.080726

Data is imbalanced so:

We will consider tha class_weight solution for this problem when we go to the modelling phase (We can use under sampling or over sampling but because of the huge difference between both classes, class weight is the best solution)

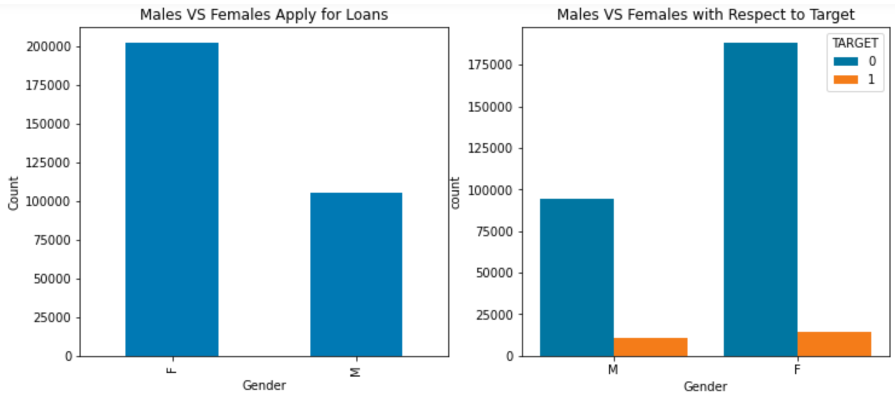
Accuracy is not a proper evaluation metric, it will be misleading so we will need to check another metric (e.g. roc_auc).

Class weight is a technique that can be used to address this issue in the context of classification models.When training a model on imbalanced data, the model may become biased towards the majority class, and may have difficulty accurately predicting the minority class. To address this, class weight assigns a higher weight to the minority class and a lower weight to the majority class during training. This means that the model will pay more attention to the minority class, and will try to improve its predictions for that class.

Next we move on to Bivariate analysis:

We try to answer some questions using the data.

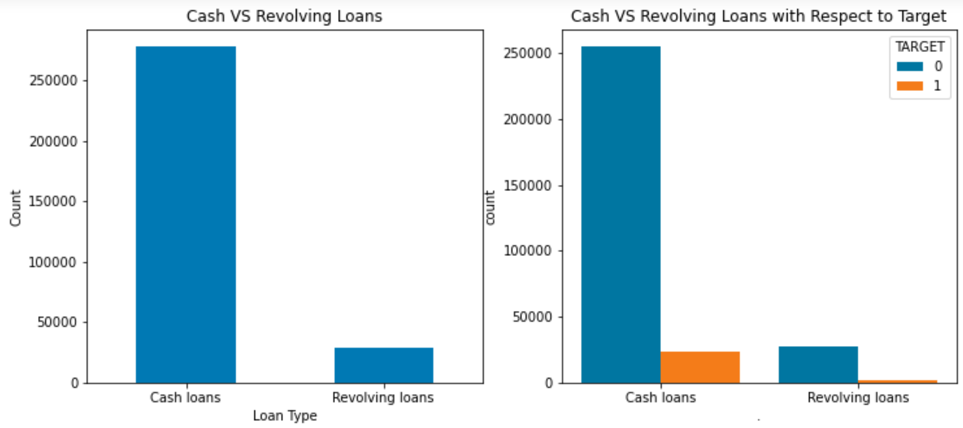
1. **Does the gender indicate anything about the target?**




```
F    0.658356
M    0.341644
Name: CODE_GENDER, dtype: float64
----------------------------------------
TARGET  CODE_GENDER
0       F              0.666046
        M              0.333954
1       F              0.570778
        M              0.429222
Name: CODE_GENDER, dtype: float64
```

Females apply for loans more than males. Gender does not affect Target variable because the difference between the gender is only 3 %.

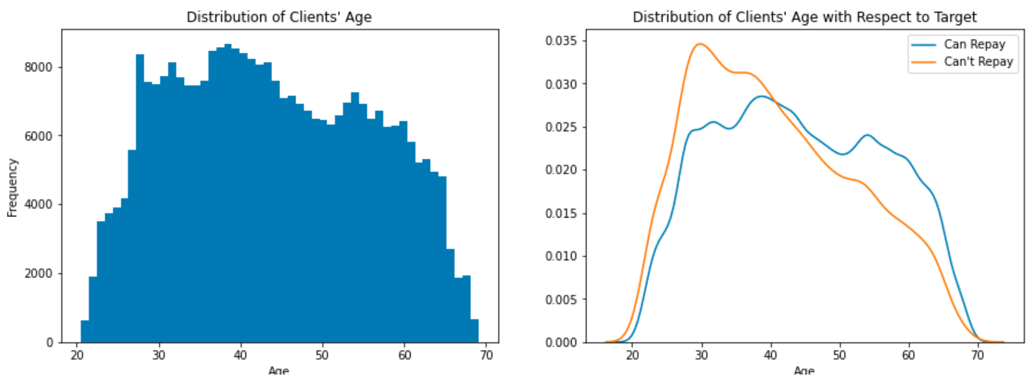
2. **Which type of loan contract clients apply for more?**




```
Cash loans         0.904787
Revolving loans    0.095213
Name: NAME_CONTRACT_TYPE, dtype: float64
----------------------------------------
TARGET  NAME_CONTRACT_TYPE
0       Cash loans            0.902100
        Revolving loans       0.097900
1       Cash loans            0.935385
        Revolving loans       0.064615
Name: NAME_CONTRACT_TYPE, dtype: float64
```

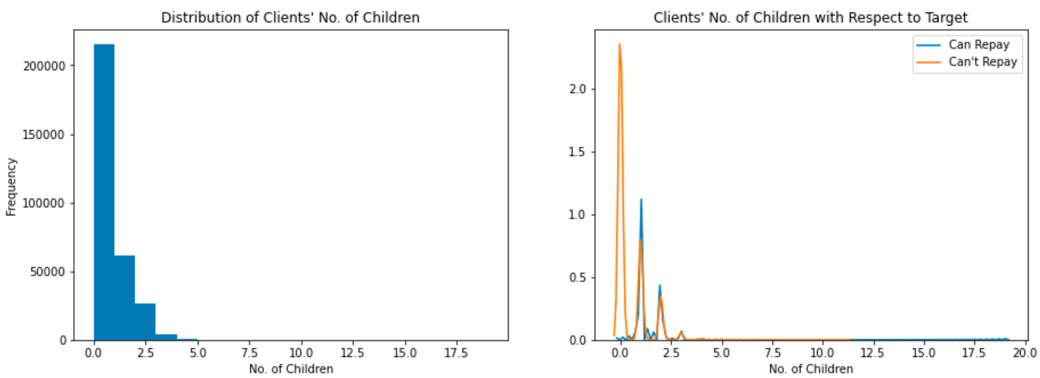
Most Clients take cash loans and it is clear that this feature has no impact on the target variable.

3. Is there a relation between the age and the ability to repay?



Clients aged about 30 years are more likely to have difficulties with repay where those aged about 40 and more can repay well. This feature is going to be important for our model.

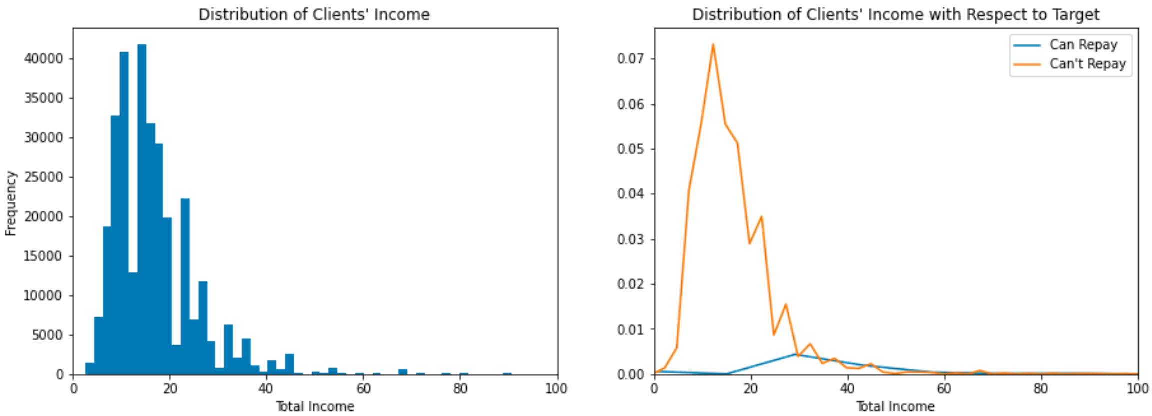
4. Does the no of children of the client affect the ability to repay?

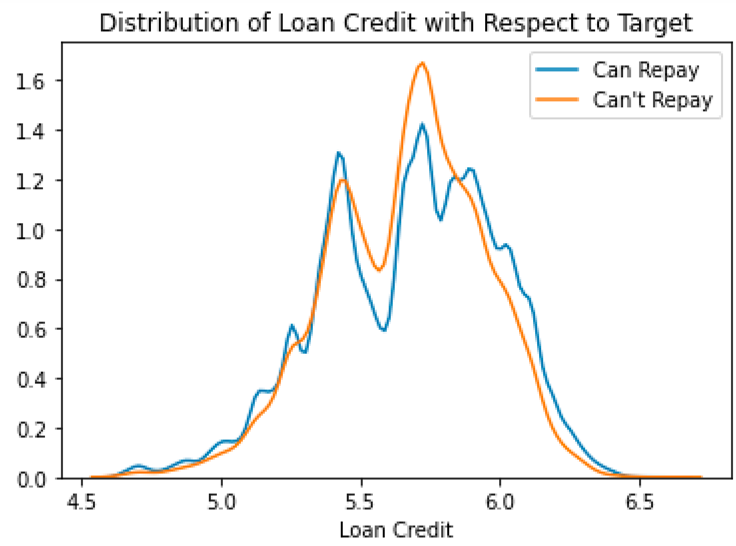


Clients without any children apply for loans more than others and with increasing number of children clients don't tend to apply for loan.

5. **Is there a relationship between client income and the amount of loan they apply for? Does income and credit affect in the ability to repay?**

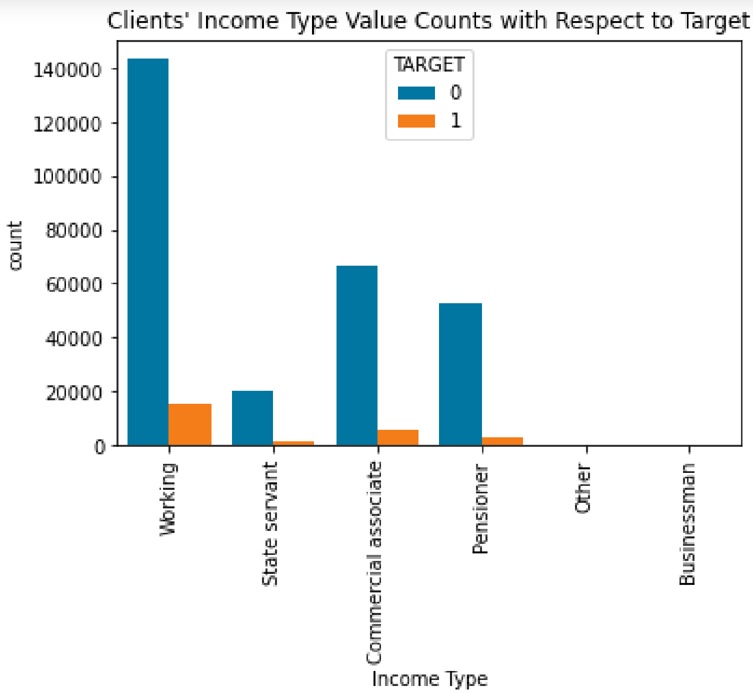
```
plt.show()
```





* Client's with low income tends to apply for loans more than others with high income.
* The more client's income is, the larger loan amount apply for.
* Client't with income more than 3M tends always to repay, so this feature may help in our target (Frist Graph on the right)
* Clients with income between 10 and 18 are less likely to repay, vice versa. (Frist Graph on the right)
* We can see that Loan Credit isn't affected by the Target Distirbution.

6. **What's most income type of clients?**



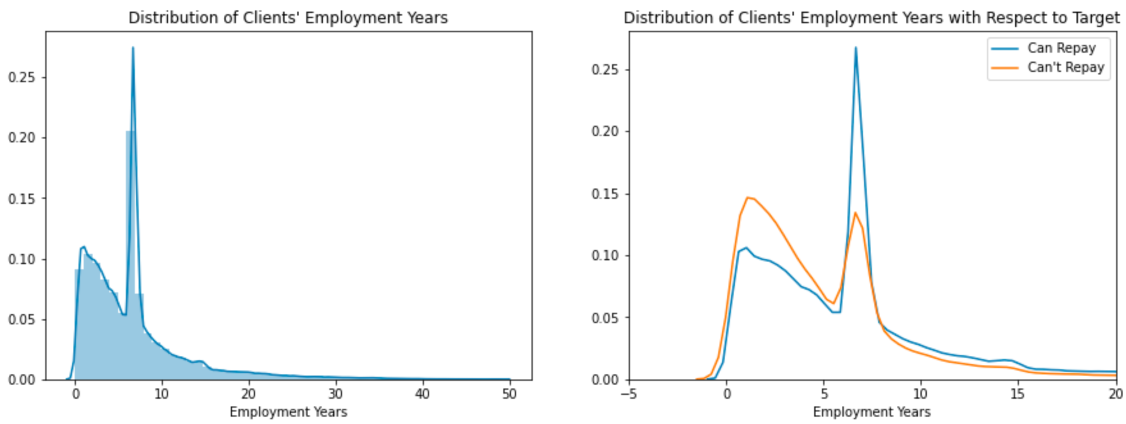
Working clients are more willing to apply for loans than others. Although a few businessmen and students apply for loans but they always repay.

7. What’s most high education degree for clients?

```
NAME_EDUCATION_TYPE            TARGET
Academic degree                0            161
                               1              3
Higher education               0          70854
                               1           4009
Incomplete higher              0           9405
                               1            872
Lower secondary                0           3399
                               1            417
Secondary / secondary special  0         198867
                               1          19523
Name: TARGET, dtype: int64
```

Clients with secondary high education level are more willing to apply for loans than others. Almost 98% of clients with Academic degree high education level can repay their loans.

8. Is there a relation between employment year and the ability to repay?



The peek in the graph is because of imputation. Clients with employment years less than 5 years tend to apply for loans more than others and they are less likely to repay especially less than 2 years.

Next we try to check statistical dependency for each variable with the target variable using chi square test and find that all variables are dependent at 5% significance level.
Next we move to check for correlation between the variables.

```
: TARGET                          1.000000
  EXT_SOURCE_2                    0.160284
  EXT_SOURCE_3                    0.157300
  DAYS_BIRTH                      0.078232
  DAYS_EMPLOYED                   0.070921
  REGION_RATING_CLIENT_W_CITY     0.060894
  REGION_RATING_CLIENT            0.058901
  DAYS_LAST_PHONE_CHANGE          0.055206
  DAYS_ID_PUBLISH                 0.051463
  REG_CITY_NOT_WORK_CITY          0.051001
  REG_CITY_NOT_LIVE_CITY          0.044399
  DAYS_REGISTRATION               0.041981
  AMT_GOODS_PRICE                 0.039622
  REGION_POPULATION_RELATIVE      0.037220
  LIVE_CITY_NOT_WORK_CITY         0.032524
  DEF_30_CNT_SOCIAL_CIRCLE        0.032398
  DEF_60_CNT_SOCIAL_CIRCLE        0.031404
  AMT_CREDIT                      0.030369
  HOUR_APPR_PROCESS_START         0.024173
  AMT_INCOME_TOTAL                0.020460
  CNT_CHILDREN                    0.019179
  AMT_REQ_CREDIT_BUREAU_MON       0.014791
  AMT_ANNUITY                     0.012816
  OBS_30_CNT_SOCIAL_CIRCLE        0.009453
  OBS_60_CNT_SOCIAL_CIRCLE        0.009344
  CNT_FAM_MEMBERS                 0.009298
  REG_REGION_NOT_WORK_REGION      0.006945
  AMT_REQ_CREDIT_BUREAU_QRT       0.005830
  REG_REGION_NOT_LIVE_REGION      0.005577
  AMT_REQ_CREDIT_BUREAU_YEAR      0.005526
  LIVE_REGION_NOT_WORK_REGION     0.002822
```

so now we drop certain columns that have no correlation and we impute the outliers for the continuous columns using the IQR (Inter Quantile Range) approach.

Feature Engineering:

There are two indicators which are critical in mortgage approval process.

* LTV, loan to value, ratio = Loan / Value of collateral. Which is 'AMT_CREDIT'/ 'AMT_GOODS_PRICE'.
* DTI, Debt to income, ratio = amount of all the monthly debt payments / the gross monthly income. Which is 'AMT_ANNUITY' / 'AMT_INCOME_TOTAL'
* Employed/Birth, Column represent days employed percentage
* Flag represents if he's greater than 30 or not
* Flag represents if his employment years is greater than 5 or not

Next we move to the Model Building Phase:
First we will convert all the categorical data by OrdinalEncoder into sequence of numbers.

Note: We didn't use OneHotEncoding because we will use RandomForest so it will behave with those numbers as a categories and split the trees based on them so we don't need OneHotEncoding and avoiding sparsity will be good.

Target encoding

Target encoding is a technique for encoding categorical variables in machine learning, where the categorical variable is replaced with the mean (or median) of the target variable for each category. It is also known as mean encoding, likelihood encoding, or impact encoding.

In target encoding, each category is replaced with the mean (or median) of the target variable for that category. For example, if the categorical variable is color, and the target variable is price, then the target encoding for the color category "red" would be the mean (or median) price of all the examples that have the color "red". This value is then used as a numerical representation of the "red" category in the model.

Target encoding can be useful for several reasons. First, it can capture the relationship between the categorical variable and the target variable more accurately than one hot encoding or ordinal encoding, especially when there are many categories. Second, it can reduce the dimensionality of the data, as each category is replaced with a single value. However, target encoding can also be prone to overfitting, especially when there are many categories or when the target variable is noisy.

To avoid overfitting, regularization techniques such as smoothing or cross-validation can be used. Smoothing involves adding a prior probability to the target encoding to avoid extreme values, while cross-validation involves using different subsets of the data for encoding and training to avoid leaking information from the target variable.

First we train random forest models with different hyperparameters without class weight or tuning and evaluate on our training and validation set.

```
Training & Validation ROC AUC Scores:
 ----------------------------------------
Training   roc auc score= 0.9599
Validation roc auc score= 0.7316

Training & Validation Confusion Metrices:
Training   confusion matrix:
 [[226149      0]
 [ 19299    560]]
Validation confusion matrix:
 [[56537     0]
 [ 4964     1]]
```

Thereafter we tune our model using grid search cv and include class weight .

```python
param_grid = {'criterion' : ['gini'],
              'class_weight' : ['balanced_subsample','balanced',None],
              'max_features' : ['sqrt', 'log2'],
              'n_estimators':[70,75,80,100]}

rf2 = RandomForestClassifier( max_depth=15, random_state=42)

grid_cv = RandomizedSearchCV(rf2,param_grid, cv=5,  scoring = 'roc_auc')

grid_cv.fit(X_train, y_train)

y_pred = grid_cv.predict_proba(X_val)
```

```python
roc_auc_score(y_val, y_pred[:, 1])
```

```
0.7325721780891876
```

```python
# I belive the best_params showed that class weight is better to be None because we used Target Encoding
grid_cv.best_params_
```

```
{'n_estimators': 100,
 'max_features': 'sqrt',
 'criterion': 'gini',
 'class_weight': None}
```

We obtain an roc_auc_score of 0.73257.

Finally we predict on the test data.

```python
y_test_proba
```

```
array([[0.83049432, 0.16950568],
       [0.85073981, 0.14926019],
       [0.92734208, 0.07265792],
       ...,
       [0.87946024, 0.12053976],
       [0.92797861, 0.07202139],
       [0.81965283, 0.18034717]])
```